



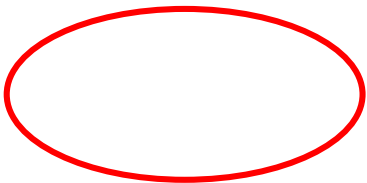
Diagrama de Classe

Agenda

01. classes



03. Métodos



02. Atributos



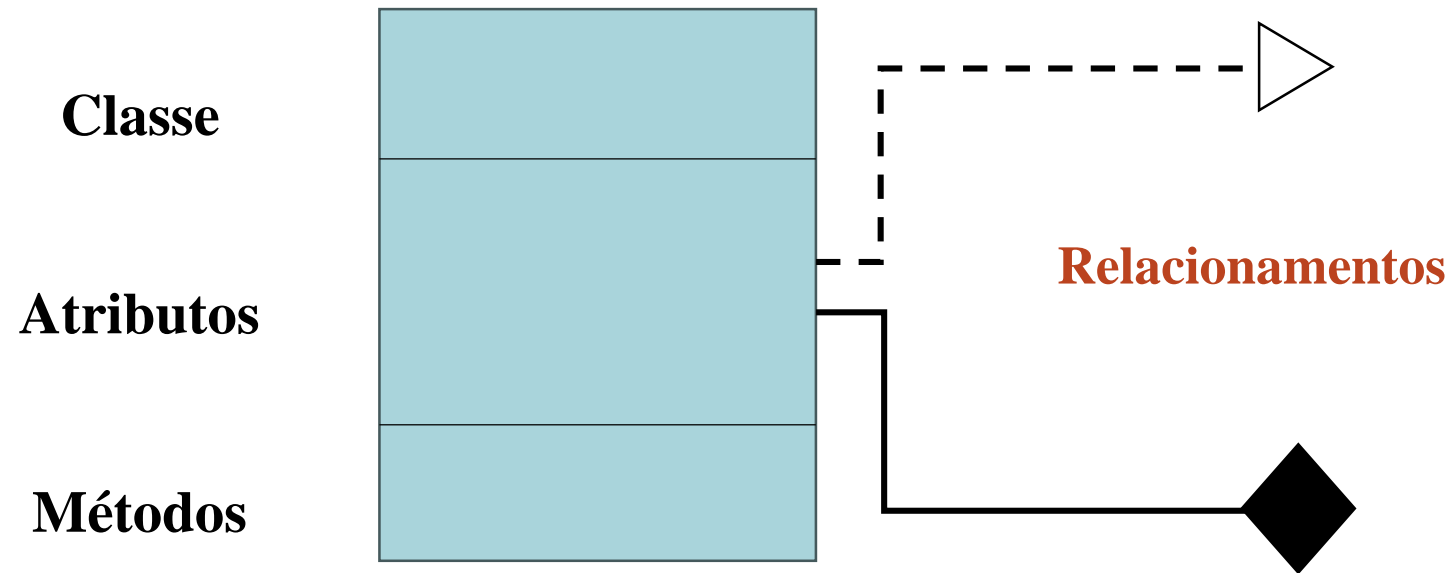
04. Relacionamentos



Introdução

Pensar em um sistema orientado a objetos é, portanto, mais do que pensar em código. É desenhar cada peça de um quebra-cabeça e pensar em como todas elas se encaixarão juntas.

Introdução



Sistema Zoológico - Classe

Vamos descrever as coisas que podem ser encontradas no sistema do zoológico, em que representamos essas coisas por classes

Classe

Animais



Instância de animal

Como você identificaria cada instância dessa classe?

Sistema Zoológico - Atributo

É um pedaço significativo de dados que contém valores que descrevem cada instância dessa classe. Também são conhecidos como campos, variáveis ou propriedades.

Para nossa classe animal podemos criar atributos como:

- nome
- núm.ident
- idade

Classe	Animais
	nome
	núm.ident
Atributos	idade



Sistema Zoológico - Atributo

É um pedaço significativo de dados que contém valores que descrevem cada instância dessa classe. Também são conhecidos como campos, variáveis ou propriedades.

Para nossa classe animal podemos criar atributos como:

- nome
- núm.ident
- idade

Classe	Animais
	-nome: string -núm.ident: int -idade: int

Atributos	
-----------	--



Sistema Zoológico - Métodos

Também conhecidos como operações ou funções, permitem especificar as características comportamentais de uma classe.

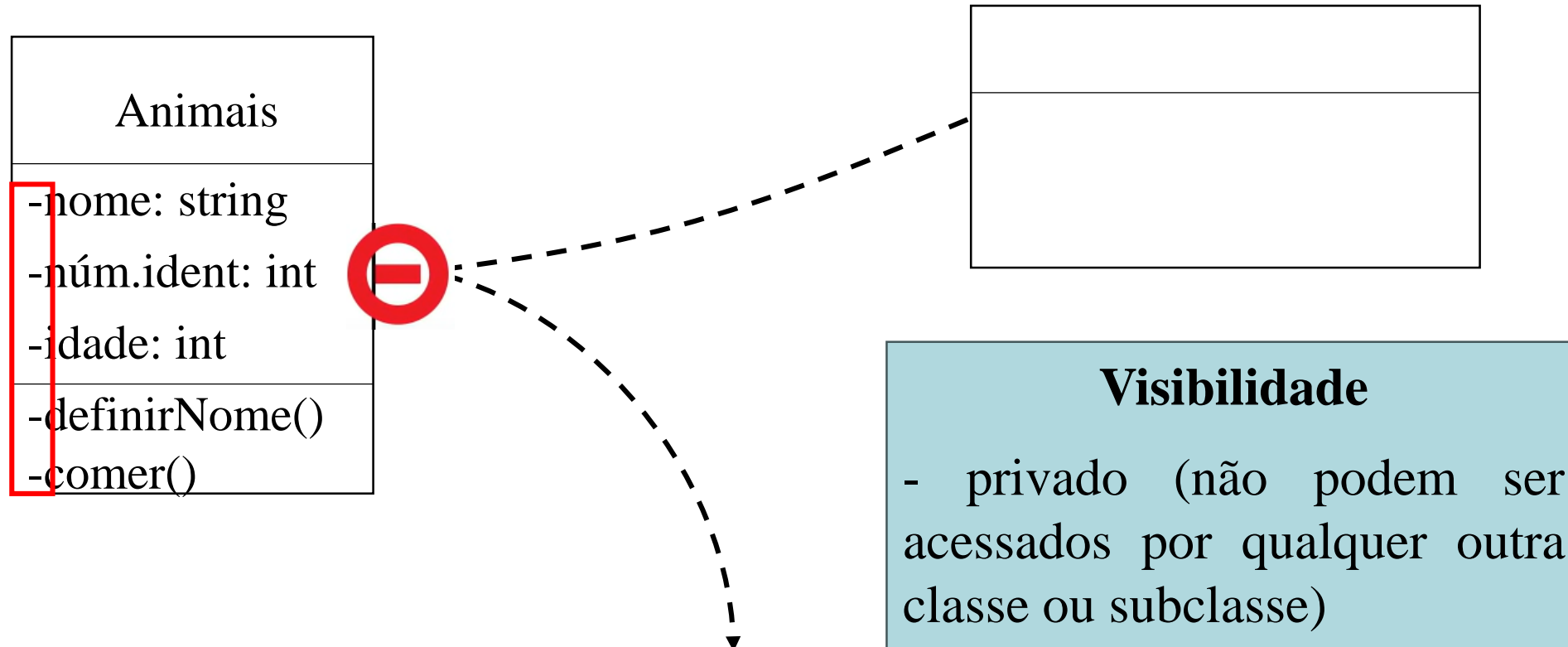
- nome
- núm.ident
- idade

Classe	Animais
Atributos	-nome: string -núm.ident: int -idade: int
Métodos	-definirNome() -comer()

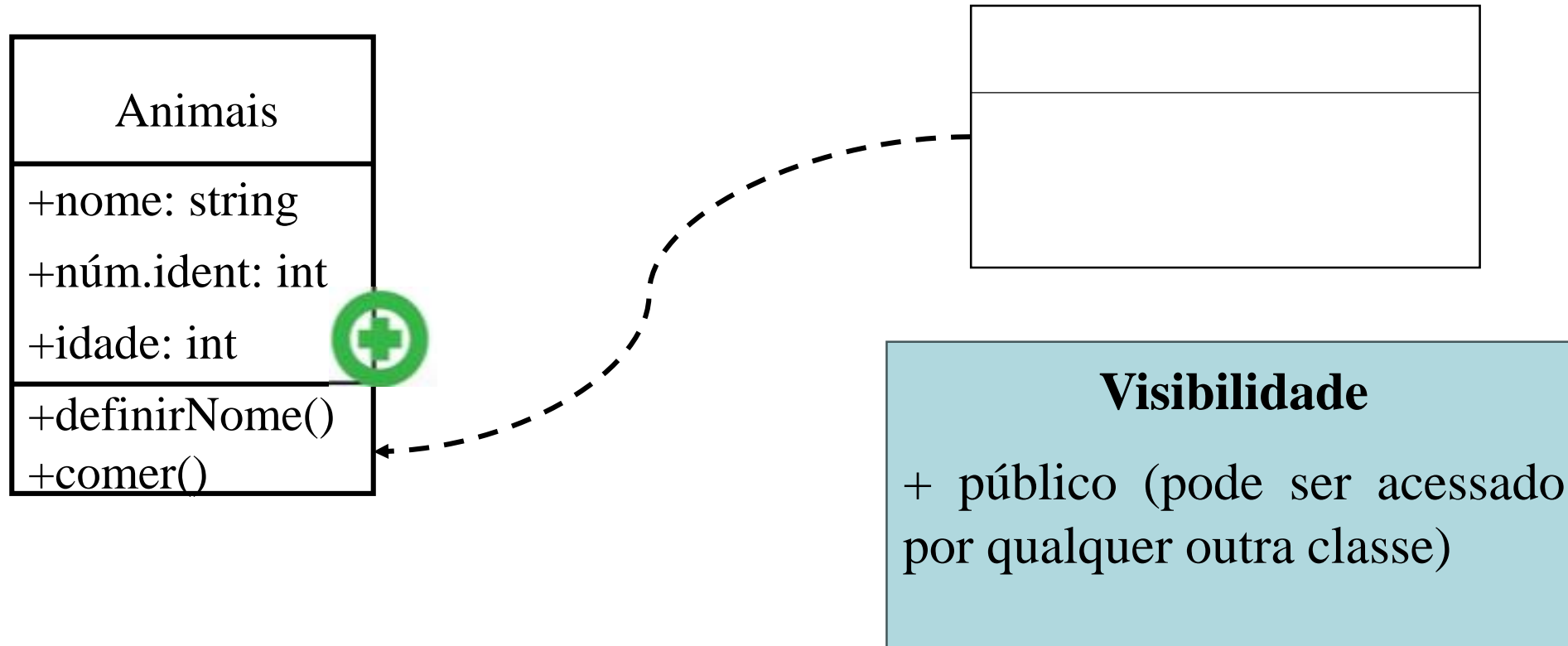


Sistema Zoológico - Visibilidade

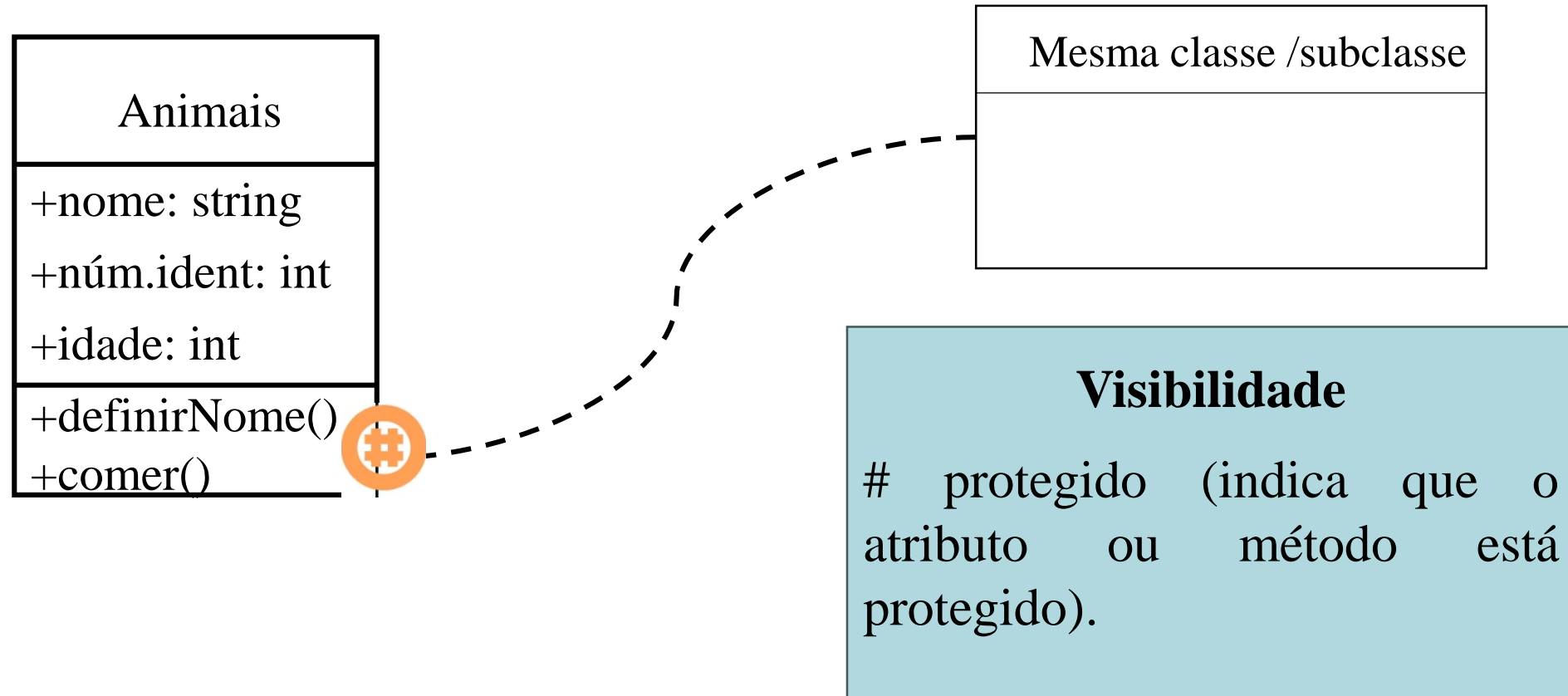
Define a acessibilidade para esse atributo ou método



Sistema Zoológico - Visibilidade

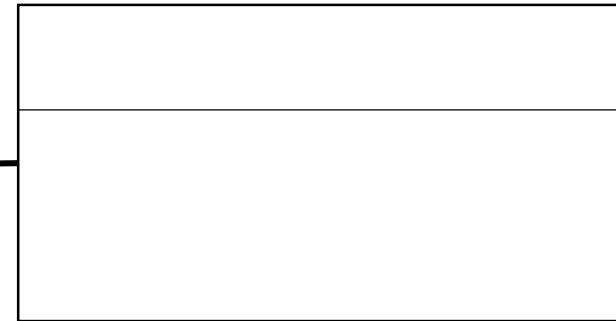
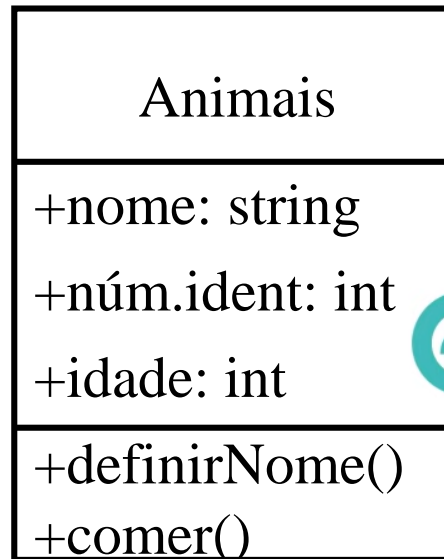


Sistema Zoológico - Visibilidade



Sistema Zoológico - Visibilidade

Pacote



Visibilidade

- privado (atributos)
- + público (métodos)
- # protegido (atributos)
- ~ pacote/padrão

Relacionamentos

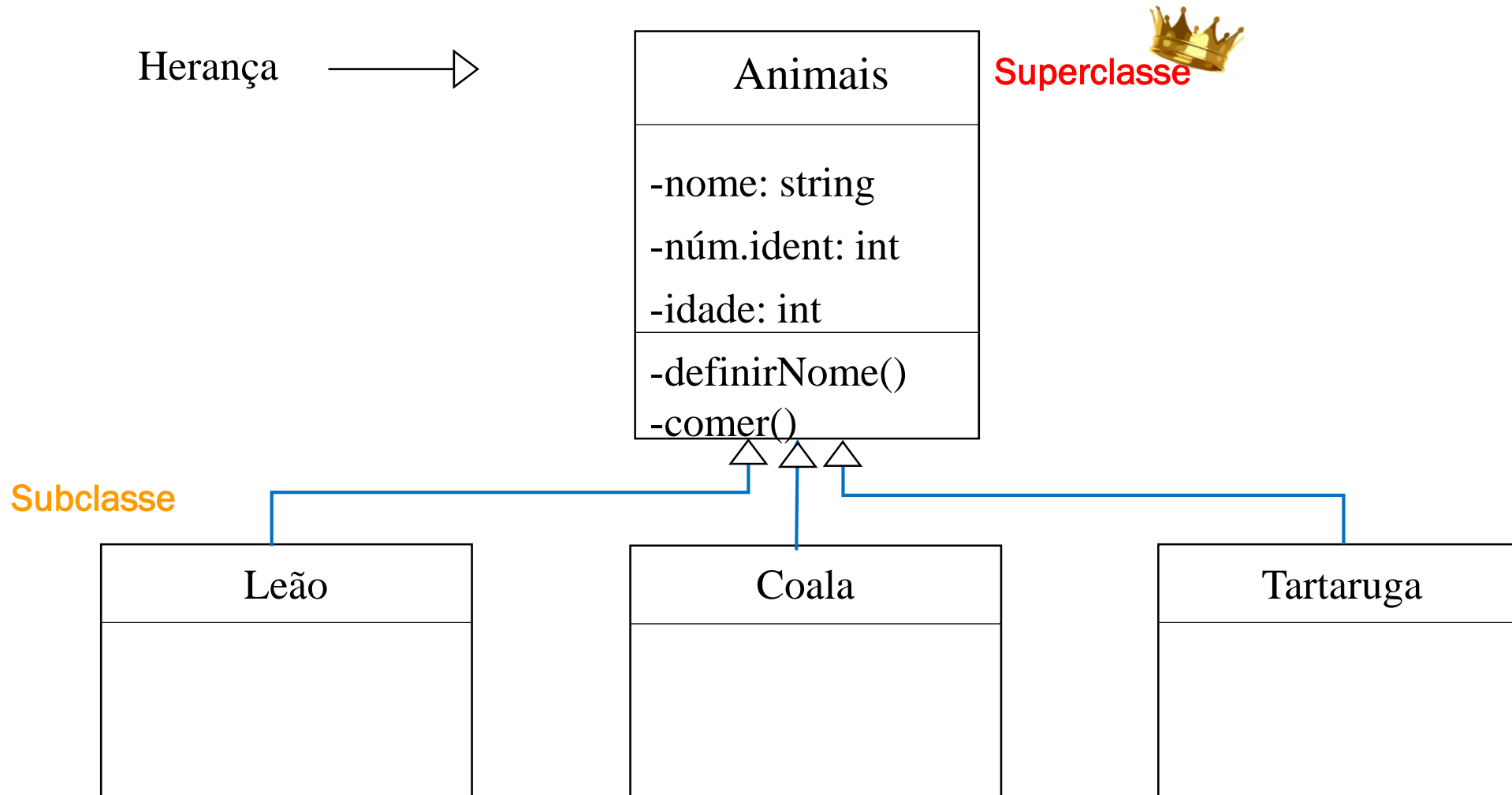
Herança —→

Animais
-nome: string -núm.ident: int -idade: int
-definirNome() -comer()

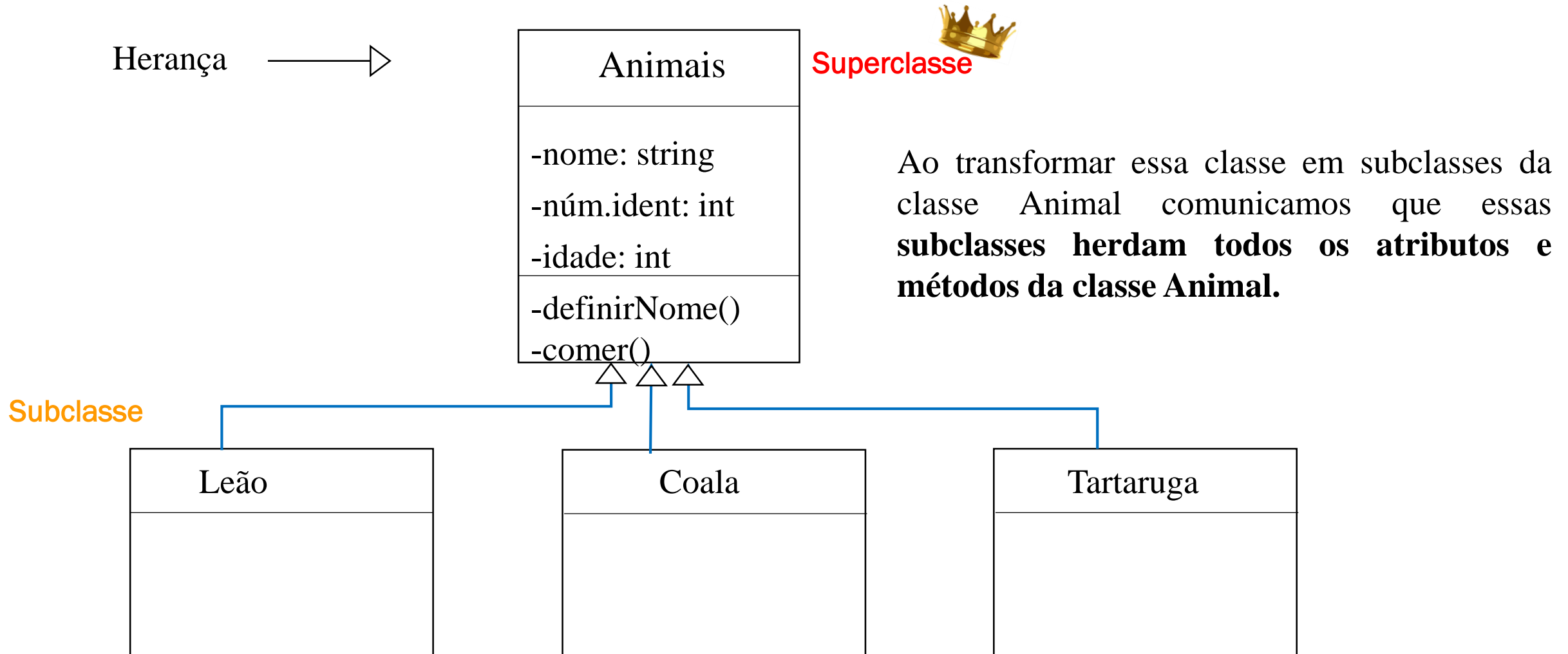


Relacionamentos - Herança

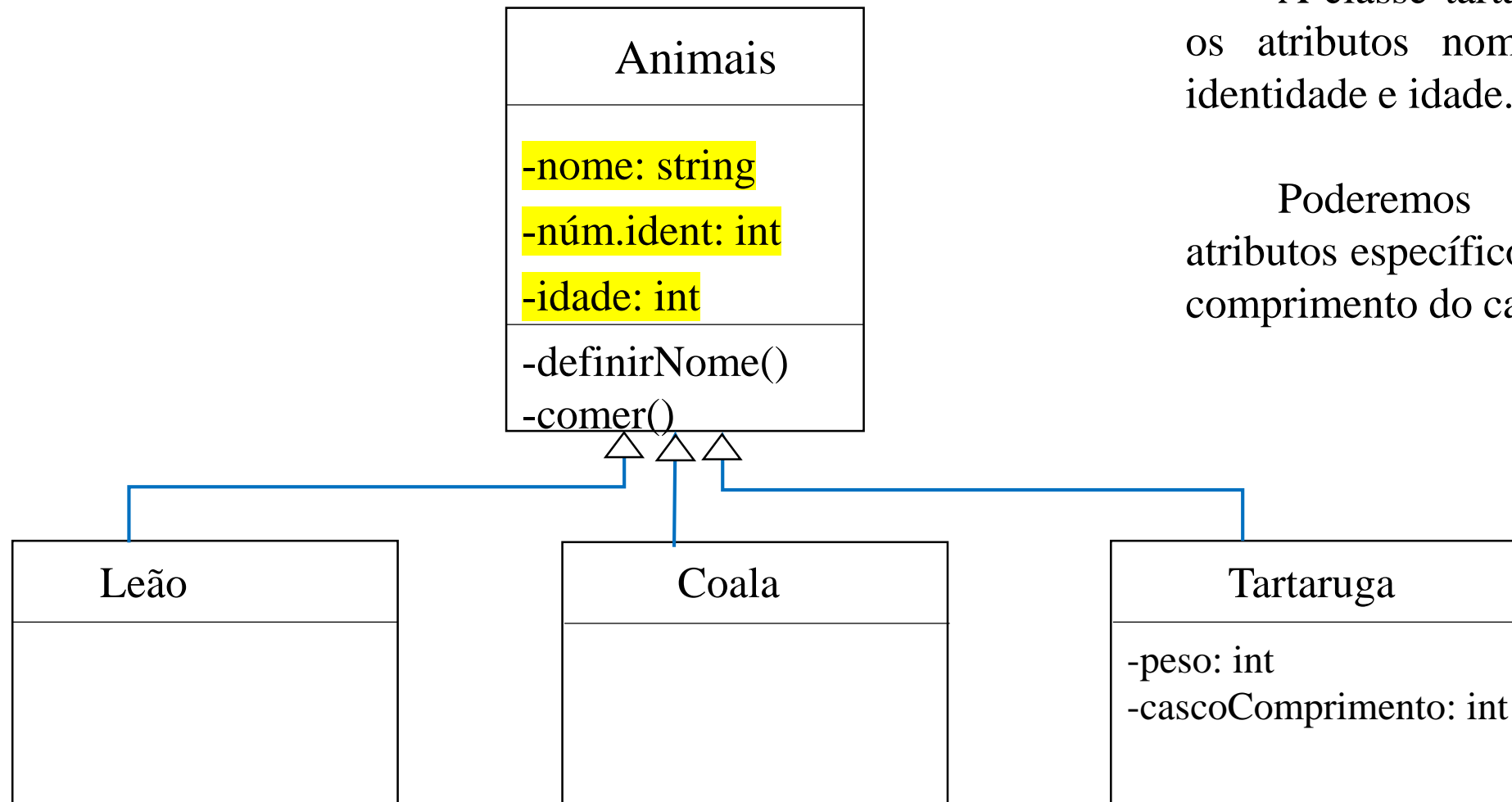
Em vez de duplicar as classes, podemos representar a relação de **herança por meio de setas abertas**.



Relacionamentos - Herança



Relacionamentos - Herança

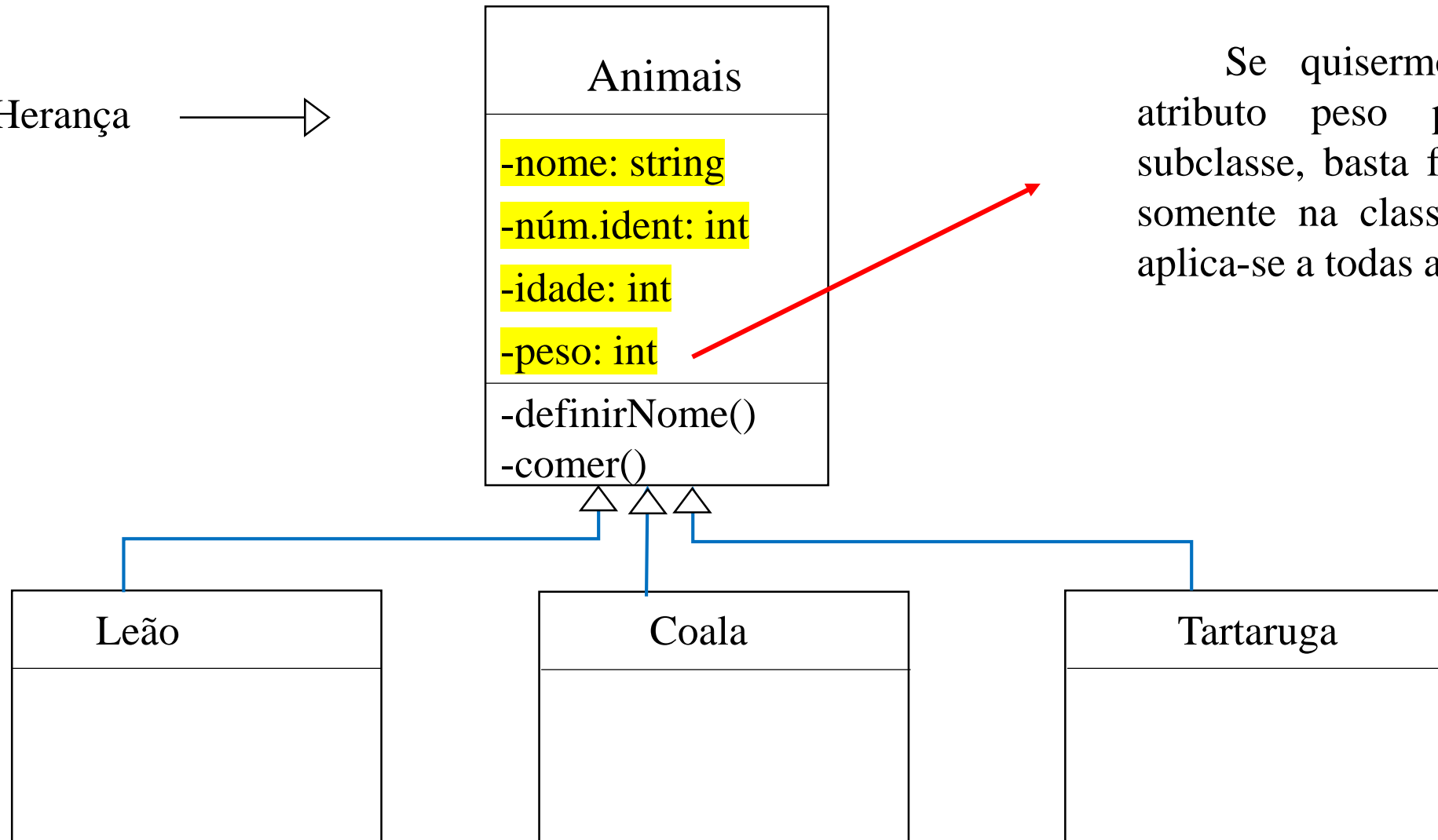


A classe tartaruga irá herdar os atributos nome, número de identidade e idade.

Poderemos adicionar atributos específicos como: peso e comprimento do casco.

Relacionamentos - Herança

Herança →



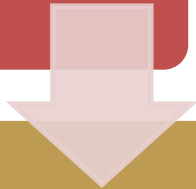
Se quisermos adicionar o atributo `peso` para todas as subclasse, basta fazer a mudança somente na classe **animal** e ela aplica-se a todas as outras classes.

Relacionamentos - Herança


- Num Sistema Bancário, construir uma classe base para Conta com atributos e métodos que serão reaproveitados em outras classes derivadas, ContaCorrente e ContaPoupança.
- Sabe-se que o método Sacar, deverá ser implementado de forma diferente em ContaCorrente.

Relacionamentos - Abstração

Ao criar uma classe para representar um objeto, como um carro ou um animal, você identifica as características essenciais desse objeto, como cor, tamanho, comportamentos etc.



Você não precisa se preocupar com todos os detalhes internos de implementação do objeto, como o funcionamento do motor do carro ou a biologia completa do animal.



Esses detalhes são abstraídos, ou seja, deixados de lado temporariamente, para se concentrar apenas nas características importantes para o contexto do problema que está sendo resolvido.

Relacionamentos - Polimorfismo

Se pensarmos na superclasse `Animal`, estabelecemos o método “emitir o som do objeto animal”, ou seja, os objetos `pato`, `cachorro` e `gato`, por exemplo, devem emitir um som ao comando do método, mas cada um fará isso de um jeito diferente.